

Chapter 15 - Putting It Altogether

Supplemental Code

Brian Fogarty

This document provides code and discussion for running a generalised ordered logit regression and for manually create smooth predicted probability lines for **confidence** and **edu** (all from the *Explaining Levels of Support for Authoritarian Regimes in Kenya* section of Chapter 15).

Generalised Ordered Logit Regression

First, we'll get the data sorted out.

```
library(tidyverse)
library(readxl)

kenya <- read_xlsx("kenya_wvs.xlsx", .name_repair =
  ~ str_sub(.x, start = 1, end = 7)) %>%
  select(starts_with("Q"))

kenya_subset <- kenya %>%
  rename(polysyst_satis = `Q252: S`,
    strong_leader = `Q235: P`,
    ideology = `Q240: L`,
    responsibility = `Q108: G`,
    confidence = `Q71: Co`,
    hhfin_satis = `Q50: Sa`,
    age = `Q262: A`,
    sex = `Q260: S`,
    edu = `Q275R: `
  ) %>%
  mutate(across(everything(), ~replace(., .x < 0, NA))) %>%
  mutate(confidence = as.numeric(fct_rev(as_factor(confidence))),
    strong_leader = as.numeric(fct_rev(as_factor(strong_leader)))
  ) %>%
  select(polysyst_satis, strong_leader, ideology, responsibility, confidence,
    hhfin_satis, age, sex, edu)

glimpse(kenya_subset)

Rows: 1,266
Columns: 9
$ polysyst_satis <dbl> 10, 4, 5, 2, 4, 7, 8, 2, 5, 4, 1, NA, 10, 5, 3, 1, 8, 1~
$ strong_leader <dbl> 3, 1, 1, 1, 1, 2, 3, 2, 4, 2, 1, NA, 4, 1, 2, 4, 4, 4, ~
$ ideology <dbl> 7, 7, 1, 5, NA, 5, 5, 6, NA, 3, 2, 1, 2, 7, 4, 3, 4, 3, ~
$ responsibility <dbl> 10, 3, 1, 1, 2, 1, 1, 5, 10, 1, 10, 10, 2, 4, 8, 1, 1, ~
$ confidence <dbl> 1, 2, 3, 4, 1, 4, 2, 1, 1, 1, 4, NA, NA, 4, 1, 1, 1, 3, ~
```

```
$ hhfin_satis    <dbl> 10, 5, 1, 6, 5, 3, 3, 6, 5, 3, 9, 1, 9, 10, 4, 1, 1, 1,~
$ age           <dbl> 63, 24, 26, 29, 37, 45, 21, 50, 26, 28, 30, 38, 40, 24,~
$ sex           <dbl> 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 2, 2, 1, 1, 2, 1, 1, 2, 2~
$ edu           <dbl> 2, 3, 3, 3, 2, 2, 2, 3, 1, 3, 1, 2, 1, 2, 3, 3, 1, 2, 2~
```

Next, we'll re-run the ordered logit regression and Brant test from the chapter.

```
library(MASS)
kenya_subset <- kenya_subset %>%
  mutate(strldr_ordfac = ordered(as_factor(strong_leader)))

summary(model.1 <- polr(strldr_ordfac ~ polysyst_satis + ideology +
  responsibility + confidence + hhfin_satis +
  age + sex + edu, method = "logistic",
  data = kenya_subset))
```

Call:

```
polr(formula = strldr_ordfac ~ polysyst_satis + ideology + responsibility +
  confidence + hhfin_satis + age + sex + edu, data = kenya_subset,
  method = "logistic")
```

Coefficients:

	Value	Std. Error	t value
polysyst_satis	0.036409	0.021426	1.6993
ideology	0.020010	0.021467	0.9321
responsibility	-0.020150	0.017609	-1.1443
confidence	0.147223	0.055628	2.6465
hhfin_satis	0.066368	0.021417	3.0989
age	0.005539	0.005525	1.0024
sex	-0.056420	0.110189	-0.5120
edu	-0.301883	0.075515	-3.9977

Intercepts:

	Value	Std. Error	t value
1 2	-1.0735	0.3887	-2.7620
2 3	0.1130	0.3874	0.2917
3 4	1.5116	0.3900	3.8759

Residual Deviance: 2979.306

AIC: 3001.306

(161 observations deleted due to missingness)

```
library(brant)
brant(model.1)
```

```
-----
Test for    X2  df  probability
-----
Omnibus      42.36  16    0
polysyst_satis  11.21   2    0
ideology      3.73   2   0.15
responsibility  2.28   2   0.32
confidence    6.64   2   0.04
hhfin_satis   6.46   2   0.04
```

```
age      0    2    1
sex      4.18   2  0.12
edu      2.7 2   0.26
```

H0: Parallel Regression Assumption holds

We see that the p -value for the Omnibus is below 0.05 and thus we violate PRA. We also see that the troublemakers are `polsyst_satis`, `confidence`, and `hhfin_satis`. Therefore, we can't trust our model estimates, particularly for these three predictors.

We can run a generalised ordered logit where we relax the PRA for the three variables that violated PRA. (Although, we could, like most people, just ignore PRA and still use ordered logit.) This model is technically called a *partial proportional odds* model because we relax PRA for certain predictors. To run the model, we'll use the `vglm()` function from the `VGAM` package; a package that does a whole bunch of things. The code is similar to what is used with `polr()` and `glm()`. For ordered logit, we need to specify `family = cumulative()` in the function.

Let's first re-run the regular ordered logit model using the `vglm()` function. To do this, we'll include `parallel = TRUE` (i.e., parallel lines are assumed for all predictors) and also `reverse = TRUE` (otherwise the signs of the coefficients will be flipped from what we found using the `polr()` function.)

```
library(VGAM)

summary(model.1a <- vglm(strldr_ordfac ~ polsyst_satis + ideology +
  responsibility + confidence + hhfin_satis +
  age + sex + edu,
  family = cumulative(parallel = TRUE, reverse = TRUE),
  data = kenya_subset))
```

Call:

```
vglm(formula = strldr_ordfac ~ polsyst_satis + ideology + responsibility +
  confidence + hhfin_satis + age + sex + edu, family = cumulative(parallel = TRUE,
  reverse = TRUE), data = kenya_subset)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)	
(Intercept):1	1.073480	0.379208	2.831	0.00464	**
(Intercept):2	-0.112997	0.377387	-0.299	0.76462	
(Intercept):3	-1.511647	0.380244	-3.975	7.02e-05	***
polsyst_satis	0.036408	0.020957	1.737	0.08233	.
ideology	0.020010	0.020841	0.960	0.33700	
responsibility	-0.020150	0.017018	-1.184	0.23640	
confidence	0.147223	0.055167	2.669	0.00761	**
hhfin_satis	0.066368	0.020964	3.166	0.00155	**
age	0.005539	0.005532	1.001	0.31674	
sex	-0.056418	0.109993	-0.513	0.60800	
edu	-0.301881	0.075252	-4.012	6.03e-05	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Names of linear predictors: `logitlink(P[Y>=2])`, `logitlink(P[Y>=3])`,
`logitlink(P[Y>=4])`

Residual deviance: 2979.306 on 3304 degrees of freedom

Log-likelihood: -1489.653 on 3304 degrees of freedom

Number of Fisher scoring iterations: 5

No Hauck-Donner effect found in any of the estimates

Exponentiated coefficients:

polysyst_satis	ideology	responsibility	confidence	hhfin_satis
1.0370791	1.0202111	0.9800521	1.1586122	1.0686199
age	sex	edu		
1.0055540	0.9451435	0.7394258		

We see the same results we attained using the `polr()` function.¹

Now, let's run the partial proportional odds version. To do this, we include `parallel = FALSE ~ polysyst_satis + confidence + hhfin_satis` in the `cumulative()` function; including the three variables specifies that we want the partial proportional odds only for them. (If we want to relax the PRA for all the predictors, we would include `parallel = FALSE` or just leave out that part entirely).

```
summary(model.1b <- vglm(strldr_ordfac ~ polysyst_satis + ideology +
  responsibility + confidence + hhfin_satis +
  age + sex + edu,
  family = cumulative(parallel = FALSE ~ polysyst_satis +
    confidence + hhfin_satis, reverse = TRUE),
  data = kenya_subset))
```

Call:

```
vglm(formula = strldr_ordfac ~ polysyst_satis + ideology + responsibility +
  confidence + hhfin_satis + age + sex + edu, family = cumulative(parallel = FALSE ~
  polysyst_satis + confidence + hhfin_satis, reverse = TRUE),
  data = kenya_subset)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)	
(Intercept):1	0.914292	0.409367	2.233	0.025521	*
(Intercept):2	-0.172742	0.388411	-0.445	0.656509	
(Intercept):3	-1.317501	0.407988	-3.229	0.001241	**
polysyst_satis:1	0.063963	0.029643	2.158	0.030948	*
polysyst_satis:2	0.080792	0.023819	3.392	0.000694	***
polysyst_satis:3	-0.025344	0.026497	-0.957	0.338819	
ideology	0.019517	0.020922	0.933	0.350919	
responsibility	-0.018215	0.017078	-1.067	0.286145	
confidence:1	0.059979	0.076864	0.780	0.435193	
confidence:2	0.132464	0.062420	2.122	0.033827	*
confidence:3	0.236134	0.070983	3.327	0.000879	***
hhfin_satis:1	0.126548	0.030660	4.128	3.67e-05	***
hhfin_satis:2	0.049145	0.023804	2.065	0.038967	*

¹The somewhat strange Hauck-Donner Effect refers to whether any of the estimates are near the boundary space (i.e., whether close to 0 or 1) in the ML estimation. Technically, when this occurs one should treat the estimate(s) as possibly suspect.

```

hhfin_satis:3      0.045888    0.026418    1.737 0.082388 .
age                0.005093    0.005551    0.918 0.358851
sex               -0.063601    0.110378   -0.576 0.564476
edu               -0.302151    0.075515   -4.001 6.30e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Names of linear predictors: logitlink(P[Y>=2]), logitlink(P[Y>=3]),
logitlink(P[Y>=4])

Residual deviance: 2950.516 on 3298 degrees of freedom

Log-likelihood: -1475.258 on 3298 degrees of freedom

Number of Fisher scoring iterations: 6

No Hauck-Donner effect found in any of the estimates

```

Exponentiated coefficients:

polsyst_satis:1	polsyst_satis:2	polsyst_satis:3	ideology	responsibility
1.0660529	1.0841451	0.9749744	1.0197082	0.9819495
confidence:1	confidence:2	confidence:3	hhfin_satis:1	hhfin_satis:2
1.0618146	1.1416383	1.2663446	1.1349040	1.0503724
hhfin_satis:3	age	sex	edu	
1.0469567	1.0051063	0.9383798	0.7392267	

Notice that there is a coefficient for each predictor for $J - 1$ categories of the outcome variable for the three predictors specified in the `cumulative()` function - `polsyst_satis`, `confidence`, and `hhfin_satis`. Why 3? Like unordered outcome regression models, the effects are estimated by comparing categories in the outcome variable. Since there are 4 values in the outcome variable, we get 3 different coefficients.

Let's look at the results for `confidence`. The first coefficient, `confidence:1`, is the effect of confidence in the national government for the lowest level of `strldr_ordfac` compared to the higher **3** levels of `strldr_ordfac`. The second coefficient, `confidence:2`, is the effect of confidence in the national government for the lowest **2** levels of `strldr_ordfac` compared to the higher **2** levels of `strldr_ordfac`. The last coefficient, `confidence:3`, is the effect of confidence in the national government for the lowest **3** levels of `strldr_ordfac` compared to the highest level of `strldr_ordfac`.

To understand this better, let's perform odds ratio interpretations of the `confidence` effects; notice that the `vglm()` function provides the exponentiated coefficients (i.e., odds ratio) at the bottom of the output.

For `confidence:1`, we see that the odds ratio is 1.06. We interpret this as *for a one-unit increase in confidence in the national government, the odds of believing a strong leader is 'fairly bad', 'fairly good' or 'very good' increase by a factor of 1.06 (or, 6%)*. Hence, the effect is for these three categories versus the first category of `strldr_ordfac` ('very bad').

For `confidence:2`, the odds ratio value is 1.14. We interpret this as *for a one-unit increase in confidence in the national government, the odds of believing a strong leader is 'fairly good' or 'very good' increase by a factor of 1.14 (or, 14%)*. Hence, the effect is for these two categories versus the first two categories of `strldr_ordfac` ('very bad', 'fairly bad').

The odds ratio value for `confidence:3` is 1.27. We interpret this as *for a one-unit increase in confidence in the national government, the odds of believing a strong leader is 'very good' increases by a factor of 1.27 (or, 27%)*. Hence, the effect is the highest category versus the first three categories of `strldr_ordfac` ('very bad', 'fairly bad', 'fairly good').

Interpreting the three coefficients for `polsyst_satis` and `hhfin_satis` are similar.
As should be obvious, generalised ordered outcome models can be a beast.

Manually Creating Predicted Probability Plots

As noted in the chapter, the predicted probability plots for `confidence` and `edu` (i.e., when they are on the x -axis) are choppy and not great aesthetically. This is due to using the `geom_line()` and `ggpredict()` functions on variables with only a few values. To make the lines smoother, we can manually create predicted probabilities for the plot instead of using `ggpredict()`. The trade-off is that it requires a lot more code. First, we'll create a plot with `confidence` on the x -axis and then we'll do one with `edu` on the x -axis.

Confidence Predicted Probability Plot

First, we will create a new dataset that includes the predicted probabilities for `strldr_ordfac` across the values of `confidence` with the rest of the predictors set at their means. We are going to create 100 values between 1 and 4 (the categories of `confidence`) for each level of `strldr_ordfac` (which has 4 levels); so, 400 values total ($4 * 100 = 400$). Having 100 values for each level allow the curves to look smooth in the plot.

We will use the `rep()` function to attain our main values. The `rep()` function replicates the values for the sequence (`seq()`) 1 to 4 for `confidence`. We attain the 100 values with `length.out=100`. The 4 in the code specifies to do it four times and corresponds to the number of values in the variable we are using for the predicted probability lines. Here it is 4 because there are four values in `strldr_ordfac`. So, the replication is done for when `strldr_ordfac=1`, then for when `strldr_ordfac=2`, etc. We'll use the `with()` function to add these new variables to our existing data object (`kenya_subset`) and save the new data frame as `ppdata`.

```
ppdata <- with(kenya_subset,
  data.frame(confidence = rep(seq(from = 1, to = 4, length.out = 100), 4),
    polysyst_satis = mean(polysyst_satis, na.rm=TRUE),
    ideology = mean(ideology, na.rm=TRUE),
    responsibility = mean(responsibility, na.rm=TRUE),
    hhfin_satis = mean(hhfin_satis, na.rm=TRUE),
    age = mean(age, na.rm=TRUE),
    sex = mean(sex, na.rm=TRUE),
    edu = mean(edu, na.rm=TRUE)))
```

```
head(ppdata)
```

	confidence	polysyst_satis	ideology	responsibility	hhfin_satis	age
1	1.000000	4.689297	5.50945	4.654459	4.87619	30.74027
2	1.030303	4.689297	5.50945	4.654459	4.87619	30.74027
3	1.060606	4.689297	5.50945	4.654459	4.87619	30.74027
4	1.090909	4.689297	5.50945	4.654459	4.87619	30.74027
5	1.121212	4.689297	5.50945	4.654459	4.87619	30.74027
6	1.151515	4.689297	5.50945	4.654459	4.87619	30.74027
	sex	edu				
1	1.494043	1.933333				
2	1.494043	1.933333				
3	1.494043	1.933333				
4	1.494043	1.933333				
5	1.494043	1.933333				
6	1.494043	1.933333				

```
tail(ppdata)
```

	confidence	polysyst_satis	ideology	responsibility	hhfin_satis	age
395	3.848485	4.689297	5.50945	4.654459	4.87619	30.74027
396	3.878788	4.689297	5.50945	4.654459	4.87619	30.74027
397	3.909091	4.689297	5.50945	4.654459	4.87619	30.74027

```

398  3.939394      4.689297  5.50945      4.654459      4.87619 30.74027
399  3.969697      4.689297  5.50945      4.654459      4.87619 30.74027
400  4.000000      4.689297  5.50945      4.654459      4.87619 30.74027
      sex      edu
395 1.494043 1.933333
396 1.494043 1.933333
397 1.494043 1.933333
398 1.494043 1.933333
399 1.494043 1.933333
400 1.494043 1.933333

```

This shows that the created values start with `confidence = 1` and then increases based on what we asked it do. Specifically, it takes the range of `confidence` and divides it by 100, while keeping the rest of the variables at their means.

Next, we will attain new predictions for `model.1` (from earlier in this document) using our new data (`ppdata`). Then we combine these predictions with our data `ppdata` using the `cbind()` function.

```

ppdata1 <- cbind(ppdata, predict(model.1, ppdata, type="probs"))
head(ppdata1)

```

```

      confidence polsyst_satis ideology responsibility hhfin_satis      age
1    1.000000      4.689297  5.50945      4.654459      4.87619 30.74027
2    1.030303      4.689297  5.50945      4.654459      4.87619 30.74027
3    1.060606      4.689297  5.50945      4.654459      4.87619 30.74027
4    1.090909      4.689297  5.50945      4.654459      4.87619 30.74027
5    1.121212      4.689297  5.50945      4.654459      4.87619 30.74027
6    1.151515      4.689297  5.50945      4.654459      4.87619 30.74027
      sex      edu      1      2      3      4
1 1.494043 1.933333 0.2255100 0.2626511 0.3061787 0.2056602
2 1.494043 1.933333 0.2247318 0.2623147 0.3065635 0.2063900
3 1.494043 1.933333 0.2239555 0.2619765 0.3069463 0.2071217
4 1.494043 1.933333 0.2231811 0.2616365 0.3073271 0.2078553
5 1.494043 1.933333 0.2224086 0.2612948 0.3077058 0.2085908
6 1.494043 1.933333 0.2216380 0.2609514 0.3080824 0.2093282

```

```

tail(ppdata1)

```

```

      confidence polsyst_satis ideology responsibility hhfin_satis      age
395  3.848485      4.689297  5.50945      4.654459      4.87619 30.74027
396  3.878788      4.689297  5.50945      4.654459      4.87619 30.74027
397  3.909091      4.689297  5.50945      4.654459      4.87619 30.74027
398  3.939394      4.689297  5.50945      4.654459      4.87619 30.74027
399  3.969697      4.689297  5.50945      4.654459      4.87619 30.74027
400  4.000000      4.689297  5.50945      4.654459      4.87619 30.74027
      sex      edu      1      2      3      4
395 1.494043 1.933333 0.1606767 0.2247148 0.3320740 0.2825344
396 1.494043 1.933333 0.1600760 0.2242593 0.3322250 0.2834397
397 1.494043 1.933333 0.1594771 0.2238032 0.3323731 0.2843466
398 1.494043 1.933333 0.1588800 0.2233463 0.3325184 0.2852553
399 1.494043 1.933333 0.1582847 0.2228887 0.3326608 0.2861658
400 1.494043 1.933333 0.1576912 0.2224304 0.3328004 0.2870780

```

We see the predicted probabilities for each category of `strldr_ordfac` have been attached as columns (1 through 4).

For the plot, we need to reshape our data from wide-to-long format by stacking variable values. We'll use the `melt()` function from the `reshape2` package. This is an older `tidyverse` function that still works, but is superseded. We specify the data and our `id.vars` (which are left alone). Everything else is assumed to be stacked. We specify the `variable.name` as "Level" (which are the four levels of `strldr_ordfac`) and the `value.name` as `Probability` (which is the probability of each `strldr_ordfac` observation).

```
library(reshape2)
ppdata2 <- melt(ppdata1, id.vars = c("confidence", "polsyst_satis", "ideology",
                                     "responsibility", "hhfin_satis", "age", "sex", "edu"),
               variable.name = "Level", value.name="Probability")
```

```
head(ppdata2)
```

	confidence	polsyst_satis	ideology	responsibility	hhfin_satis	age
1	1.000000	4.689297	5.50945	4.654459	4.87619	30.74027
2	1.030303	4.689297	5.50945	4.654459	4.87619	30.74027
3	1.060606	4.689297	5.50945	4.654459	4.87619	30.74027
4	1.090909	4.689297	5.50945	4.654459	4.87619	30.74027
5	1.121212	4.689297	5.50945	4.654459	4.87619	30.74027
6	1.151515	4.689297	5.50945	4.654459	4.87619	30.74027

	sex	edu	Level	Probability
1	1.494043	1.933333	1	0.2255100
2	1.494043	1.933333	1	0.2247318
3	1.494043	1.933333	1	0.2239555
4	1.494043	1.933333	1	0.2231811
5	1.494043	1.933333	1	0.2224086
6	1.494043	1.933333	1	0.2216380

```
tail(ppdata2)
```

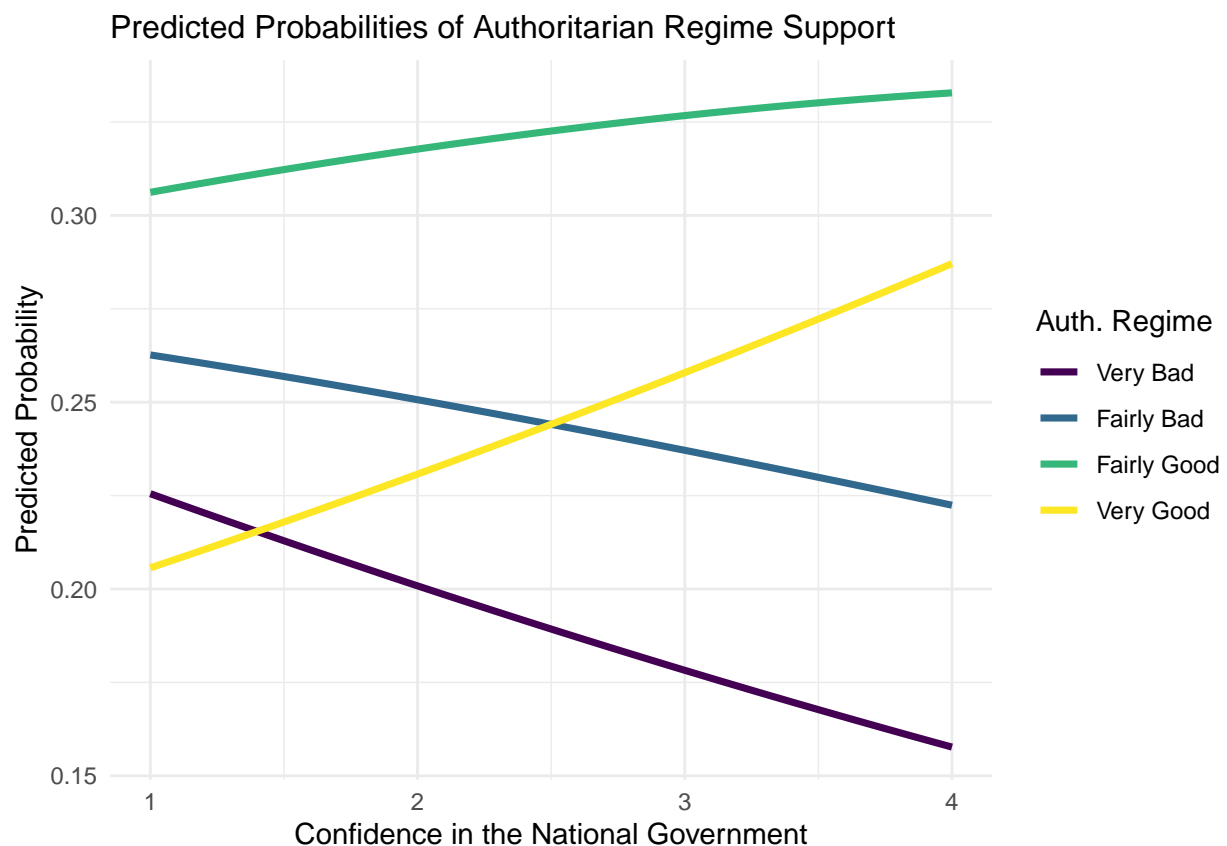
	confidence	polsyst_satis	ideology	responsibility	hhfin_satis	age
1595	3.848485	4.689297	5.50945	4.654459	4.87619	30.74027
1596	3.878788	4.689297	5.50945	4.654459	4.87619	30.74027
1597	3.909091	4.689297	5.50945	4.654459	4.87619	30.74027
1598	3.939394	4.689297	5.50945	4.654459	4.87619	30.74027
1599	3.969697	4.689297	5.50945	4.654459	4.87619	30.74027
1600	4.000000	4.689297	5.50945	4.654459	4.87619	30.74027

	sex	edu	Level	Probability
1595	1.494043	1.933333	4	0.2825344
1596	1.494043	1.933333	4	0.2834397
1597	1.494043	1.933333	4	0.2843466
1598	1.494043	1.933333	4	0.2852553
1599	1.494043	1.933333	4	0.2861658
1600	1.494043	1.933333	4	0.2870780

Because the probabilities are stacked, we now have 1600 observations instead of 400 observations. That is, for `Level = 1` (the first category of `strldr_ordfac`) there are 400 probability values corresponding to the 400 values in `confidence`. Then for `Level = 2` (the second category of `strldr_ordfac`) there are 400 probability values corresponding to the 400 values in `confidence`. Etc.

Next, we label the categories of `strldr_ordfac` (which is the `Level` variable) and plot the predicted probabilities.

```
ppdata2 %>%
  mutate(Level = recode(Level,
    `1` = "Very Bad",
    `2` = "Fairly Bad",
    `3` = "Fairly Good",
    `4` = "Very Good")) %>%
  ggplot(mapping = aes(x = confidence, y = Probability, colour = Level)) +
    geom_line(size = 1.25) +
    scale_x_continuous(limits = c(1,4), breaks = c(1:4)) +
    labs(title = "Predicted Probabilities of Authoritarian Regime Support",
      x = "Confidence in the National Government",
      y = "Predicted Probability") +
    guides(colour = guide_legend(title = "Auth. Regime"),
      fill = guide_legend(title = "Auth. Regime")) +
    theme_minimal() +
    theme(
      plot.title = element_text(size = 12)
    ) +
    scale_colour_viridis_d()
```



We now see the choppy bends at each value of `confidence` are smoothed out.

Obviously, this is a lot of work to get smooth lines and it only provides a marginal improvement over the much simpler option for `confidence` in the chapter. In uglier cases, like for `edu`, we might want to make the effort. Next, we'll replicate the above for the `edu` plot.

Education Predicted Probability Plots

We use the same code as above and just switch out `confidence` for `edu` and adjust the values, labels, etc.

```
ppdata_edu <- with(kenya_subset,
  data.frame(edu = rep(seq(from = 1, to = 3, length.out = 100), 4),
    polysyst_satis = mean(polysyst_satis, na.rm=TRUE),
    ideology = mean(ideology, na.rm=TRUE),
    responsibility = mean(responsibility, na.rm=TRUE),
    hhfin_satis = mean(hhfin_satis, na.rm=TRUE),
    age = mean(age, na.rm=TRUE),
    sex = mean(sex, na.rm=TRUE),
    confidence = mean(confidence, na.rm=TRUE)))

head(ppdata_edu)
```

	edu	polysyst_satis	ideology	responsibility	hhfin_satis	age	sex
1	1.000000	4.689297	5.50945	4.654459	4.87619	30.74027	1.494043
2	1.020202	4.689297	5.50945	4.654459	4.87619	30.74027	1.494043
3	1.040404	4.689297	5.50945	4.654459	4.87619	30.74027	1.494043
4	1.060606	4.689297	5.50945	4.654459	4.87619	30.74027	1.494043
5	1.080808	4.689297	5.50945	4.654459	4.87619	30.74027	1.494043
6	1.101010	4.689297	5.50945	4.654459	4.87619	30.74027	1.494043

```
  confidence
1    2.430868
2    2.430868
3    2.430868
4    2.430868
5    2.430868
6    2.430868

tail(ppdata_edu)
```

	edu	polysyst_satis	ideology	responsibility	hhfin_satis	age	sex
395	2.898990	4.689297	5.50945	4.654459	4.87619	30.74027	
396	2.919192	4.689297	5.50945	4.654459	4.87619	30.74027	
397	2.939394	4.689297	5.50945	4.654459	4.87619	30.74027	
398	2.959596	4.689297	5.50945	4.654459	4.87619	30.74027	
399	2.979798	4.689297	5.50945	4.654459	4.87619	30.74027	
400	3.000000	4.689297	5.50945	4.654459	4.87619	30.74027	

```
  sex confidence
395 1.494043    2.430868
396 1.494043    2.430868
397 1.494043    2.430868
398 1.494043    2.430868
399 1.494043    2.430868
400 1.494043    2.430868

ppdata_edu1 <- cbind(ppdata_edu, predict(model.1, ppdata_edu, type="probs"))
head(ppdata_edu1)
```

	edu	polysyst_satis	ideology	responsibility	hhfin_satis	age	sex
1	1.000000	4.689297	5.50945	4.654459	4.87619	30.74027	1.494043
2	1.020202	4.689297	5.50945	4.654459	4.87619	30.74027	1.494043
3	1.040404	4.689297	5.50945	4.654459	4.87619	30.74027	1.494043
4	1.060606	4.689297	5.50945	4.654459	4.87619	30.74027	1.494043

```

5 1.080808      4.689297  5.50945      4.654459      4.87619 30.74027 1.494043
6 1.101010      4.689297  5.50945      4.654459      4.87619 30.74027 1.494043
  confidence      1      2      3      4
1  2.430868 0.1510673 0.2171718 0.3341858 0.2975751
2  2.430868 0.1518511 0.2178079 0.3340390 0.2963019
3  2.430868 0.1526383 0.2184430 0.3338869 0.2950319
4  2.430868 0.1534287 0.2190769 0.3337293 0.2937650
5  2.430868 0.1542226 0.2197097 0.3335664 0.2925014
6  2.430868 0.1550197 0.2203414 0.3333980 0.2912409

```

```
tail(ppdata_edu1)
```

```

      edu polysyst_satis ideology responsibility hhfin_satis      age
395 2.898990      4.689297  5.50945      4.654459      4.87619 30.74027
396 2.919192      4.689297  5.50945      4.654459      4.87619 30.74027
397 2.939394      4.689297  5.50945      4.654459      4.87619 30.74027
398 2.959596      4.689297  5.50945      4.654459      4.87619 30.74027
399 2.979798      4.689297  5.50945      4.654459      4.87619 30.74027
400 3.000000      4.689297  5.50945      4.654459      4.87619 30.74027
      sex confidence      1      2      3      4
395 1.494043  2.430868 0.2399451 0.2684279 0.2988624 0.1927646
396 1.494043  2.430868 0.2410591 0.2688381 0.2982854 0.1918174
397 1.494043  2.430868 0.2421766 0.2692445 0.2977051 0.1908737
398 1.494043  2.430868 0.2432976 0.2696473 0.2971215 0.1899336
399 1.494043  2.430868 0.2444222 0.2700462 0.2965345 0.1889971
400 1.494043  2.430868 0.2455502 0.2704414 0.2959443 0.1880641

```

```

ppdata_edu2 <- melt(ppdata_edu1, id.vars = c("edu", "polysyst_satis", "ideology",
      "responsibility", "hhfin_satis", "age", "sex", "confidence"),
      variable.name = "Level", value.name="Probability")

```

```
head(ppdata_edu2)
```

```

      edu polysyst_satis ideology responsibility hhfin_satis      age      sex
1 1.000000      4.689297  5.50945      4.654459      4.87619 30.74027 1.494043
2 1.020202      4.689297  5.50945      4.654459      4.87619 30.74027 1.494043
3 1.040404      4.689297  5.50945      4.654459      4.87619 30.74027 1.494043
4 1.060606      4.689297  5.50945      4.654459      4.87619 30.74027 1.494043
5 1.080808      4.689297  5.50945      4.654459      4.87619 30.74027 1.494043
6 1.101010      4.689297  5.50945      4.654459      4.87619 30.74027 1.494043
  confidence Level Probability
1  2.430868      1  0.1510673
2  2.430868      1  0.1518511
3  2.430868      1  0.1526383
4  2.430868      1  0.1534287
5  2.430868      1  0.1542226
6  2.430868      1  0.1550197

```

```
tail(ppdata_edu2)
```

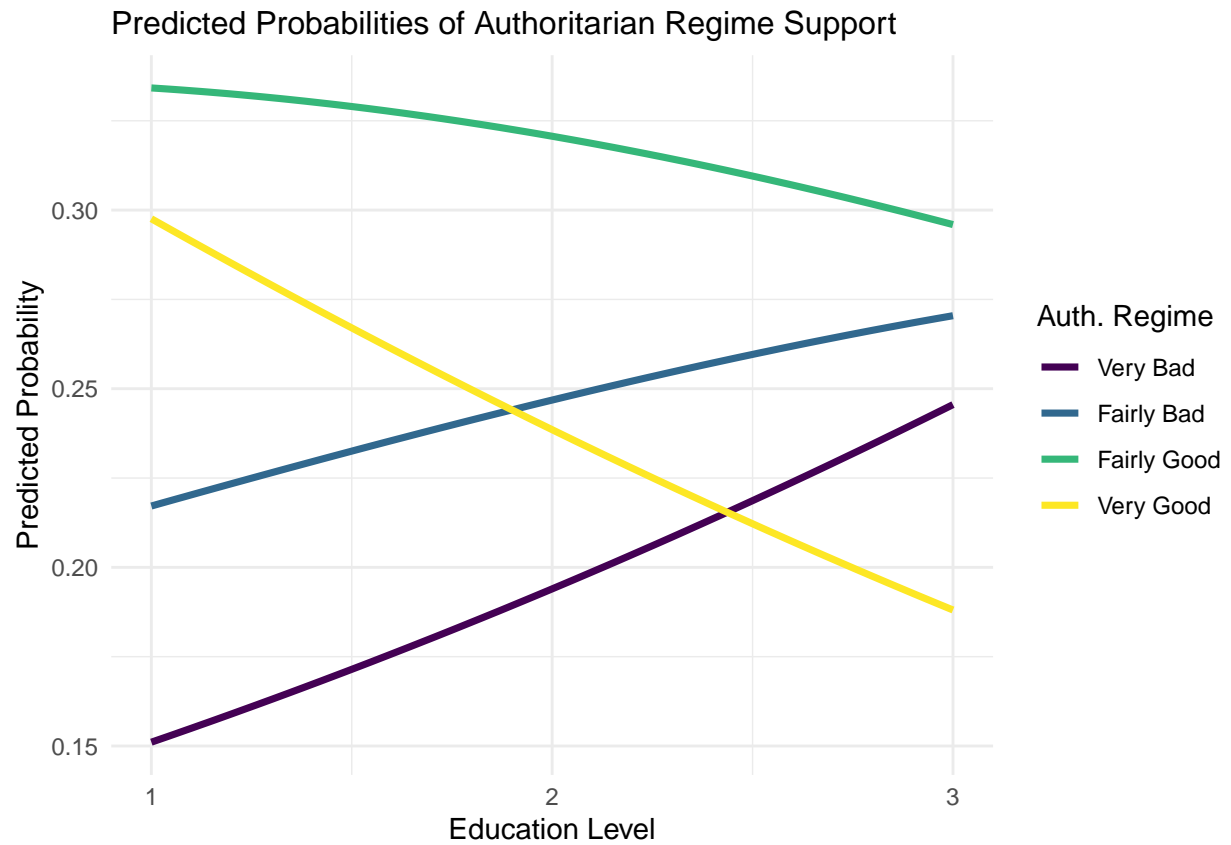
```

      edu polysyst_satis ideology responsibility hhfin_satis      age
1595 2.898990      4.689297  5.50945      4.654459      4.87619 30.74027
1596 2.919192      4.689297  5.50945      4.654459      4.87619 30.74027
1597 2.939394      4.689297  5.50945      4.654459      4.87619 30.74027
1598 2.959596      4.689297  5.50945      4.654459      4.87619 30.74027
1599 2.979798      4.689297  5.50945      4.654459      4.87619 30.74027

```

1600	3.000000	4.689297	5.50945	4.654459	4.87619	30.74027
	sex	confidence	Level	Probability		
1595	1.494043	2.430868	4	0.1927646		
1596	1.494043	2.430868	4	0.1918174		
1597	1.494043	2.430868	4	0.1908737		
1598	1.494043	2.430868	4	0.1899336		
1599	1.494043	2.430868	4	0.1889971		
1600	1.494043	2.430868	4	0.1880641		

```
ppdata_edu2 %>%
  mutate(Level = recode(Level,
    `1` = "Very Bad",
    `2` = "Fairly Bad",
    `3` = "Fairly Good",
    `4` = "Very Good")) %>%
ggplot(mapping = aes(x = edu, y = Probability, colour = Level)) +
  geom_line(size = 1.25) +
  scale_x_continuous(limits = c(1,3), breaks = c(1:3)) +
  labs(title = "Predicted Probabilities of Authoritarian Regime Support",
    x = "Education Level", y = "Predicted Probability") +
  guides(colour = guide_legend(title = "Auth. Regime"),
    fill = guide_legend(title = "Auth. Regime")) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 12)
  ) +
  scale_colour_viridis_d()
```



This is a much better looking plot for edu and worth the effort.